RAPPORT PROJET 2024-2025

Sephijin

Commanditaire: PELAY Johan, DELCEY Lucie, Sephijin

Fait par : Perrot Loick, Gomez Nolan, Treyture-Hayet Robin

TABLE DES MATIÈRES -

INTRODUCTION -

CONTEXTE

Twitch, une plateforme de streaming vidéo, génère une grande quantité de données liées aux utilisateurs, aux streamers, et aux événements en direct.
Pour ce projet nous avons du aidé un streamer du nom de Sephijin à avoir une vue d'ensemble sur ces années passés sur la plateforme de streaming. Pour qu'il puisse voir ces statistiques de façon claire et précise.

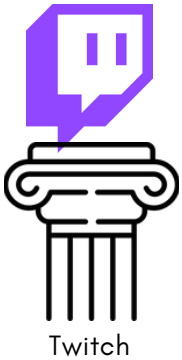
NOTRE OBJECTIFS

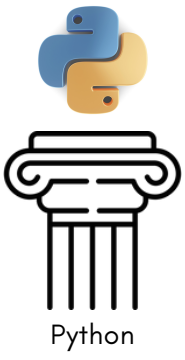
Notre objectif est de développer un outil d'analyse de données à partir de fichiers de Twitch. Il devra posséder une interface graphique qui permettra à un utilisateur, sans compétences en informatique ni en statistiques, de consulter et de comprendre les résultats.

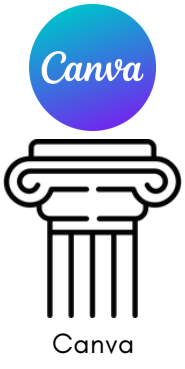
LES ETAPES

Pour le début de notre projet nous avons étudié chaque possibilité d'analyse grâce à une feuille Excel remplit de données et un site mis en place par Twitch ou nous pouvons voir les statistiques de chaque streamer. Pour réaliser cette interface graphique nous avons utilisé le langage informatique python et ces différents packages pour le réaliser. Pour ce qui est de la répartition des tâches nous avons utilisé l'application GANTT Project. Pour terminer nous avons utilisé Canva pour la rédaction du rapport.


GANTT Project

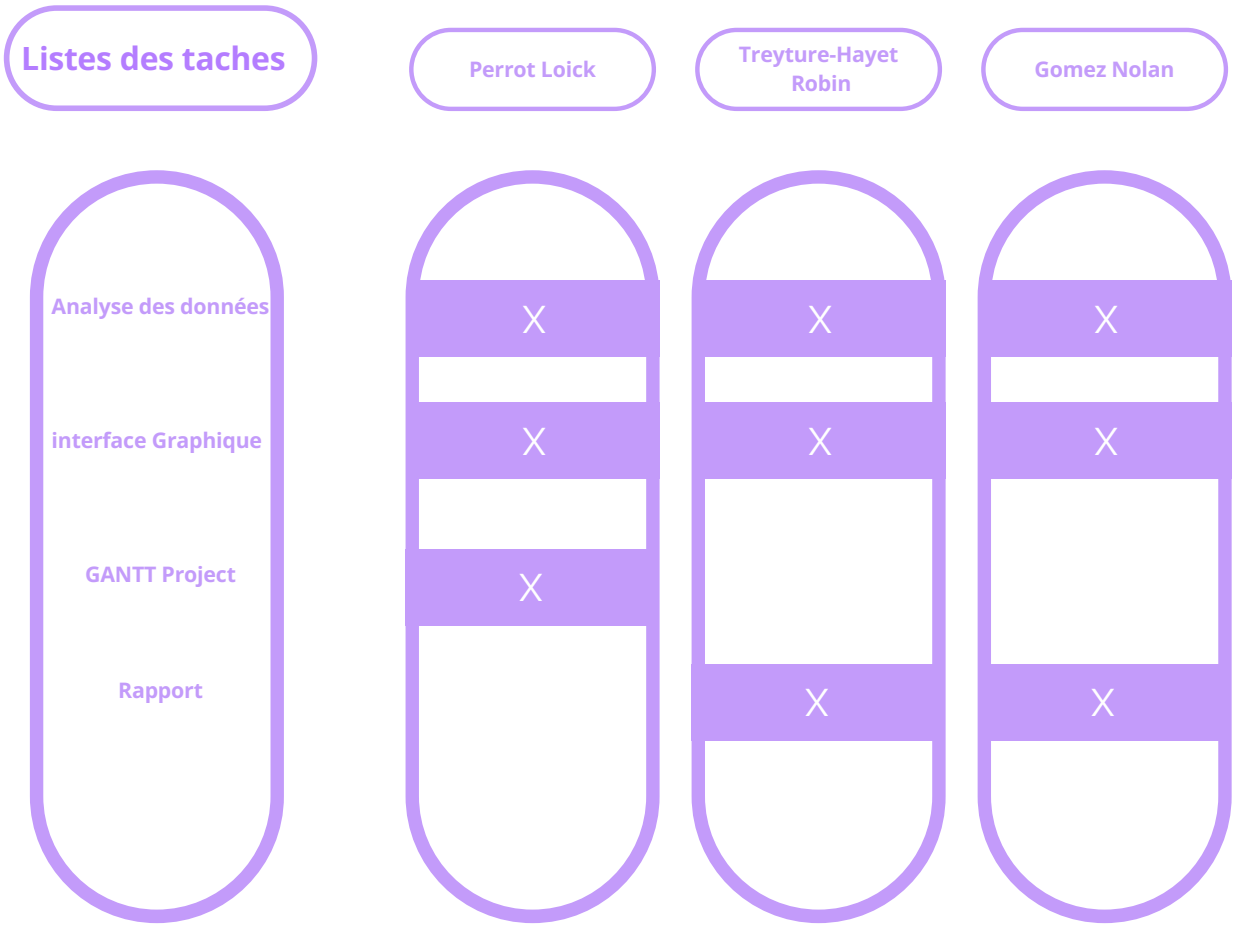

Twitch


Python



Canva

CAHIER DES CHARGES -

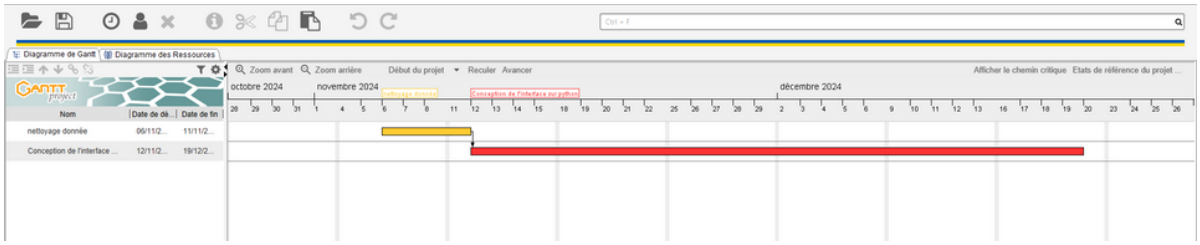


ANALYSE DES RISQUES -

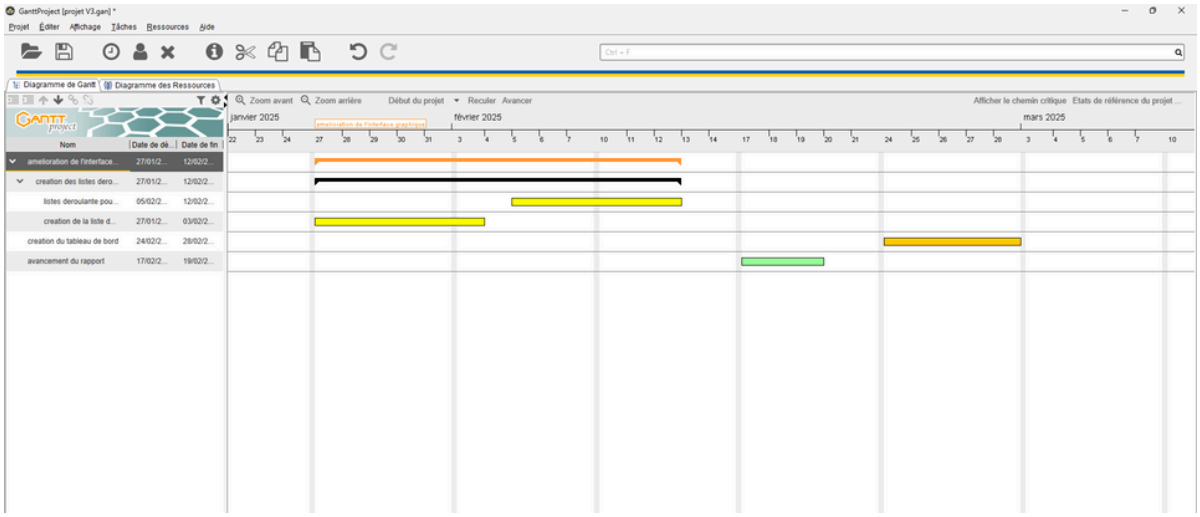
Libellé du risque	Probabilité	Impact	Cotation	Action
Perte d'un membre	1	7	7	Communication pour éviter des problèmes majeures
Perte de donnée	5	9	45	Sauvegarde régulière sur plusieurs clés USB
Mauvaise communication	2	6	12	Mettre en place des réunions au cours du projet

GANTT PROJECT -

Pour nous organiser nous avons donc utilisé l'application Gantt Project et répartit nos travaux dessus voici dessous le tout premier Gantt que nous avons créé qui a manqué de précision. Avec seulement deux activités à faire dont le nettoyage de nos données et la création de l'interface qui nous a pris la plus part de notre temps.



Pour la deuxième partie du projet nous avons un peu plus détaillé notre Gantt qui se composée de l'amélioration de notre interface avec la création de listes déroulantes pour les choix de nos variables et calcules, la finition de notre rapport et la fin de notre GANTT



ANALYSES DES DONNÉES -

Pour ce début de projet nous avons du faire face à une feuille Excel avec différentes variables.

Au final les variables que nous avons choisis et qui nous semble les plus importante à mettre en avant sont :

-Spectateurs en moyenne
-Spectateurs max.
-Nouveaux followers
-Abonnements payés au total
-Abonnements Prime
-Revenu issu des abonnements
-Revenu Prime
-Minutes passées en stream
-Date

Pour toute ces variables nous avons décidé de calculer les moyennes et les variances avec différentes fréquences Hebdomadaire, Mensuel et Trimestriel.

INTERFACE GRAPHIQUE -

Maintenant passons au sujet principale de ce projet le code python pour créer notre interface graphique et les choix que nous avons fais .

Tout d'abord voici la liste des packages et module que nous avons utilisé :

```python
import tkinter as tk
from tkinter import ttk
from tkinter.filedialog import askopenfilename
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import matplotlib.dates as mdates
```

Ensuite notre code python s'est construit en 3 parties la première partie est celle du choix et du lancement d'un fichier de type csv avec son nettoyage.

```python
# Fonction pour sélectionner un fichier CSV
def select_file():
    file_path = askopenfilename(filetypes=[("Fichiers CSV", "*.csv")], title="Choisissez un fichier CSV")
    return file_path

# Chargement du fichier CSV
file_path = select_file()
if not file_path:
    print("Aucun fichier sélectionné. Fermeture du programme.")
    exit()

# Chargement et nettoyage des données
try:
    data = pd.read_csv(file_path, encoding='utf-8')
    data.columns = data.columns.str.strip()
    data.columns = data.columns.str.replace("Â", "")  # Correction des erreurs d'encodage
    print("Colonnes disponibles :", data.columns.tolist())
    data['Date'] = pd.to_datetime(data['Date'], errors='coerce')  # Conversion en date
    data.set_index('Date', inplace=True)
except Exception as e:
    print(f"Erreur lors du chargement du fichier : {e}")
    exit()

# Vérification des colonnes nécessaires
required_columns = ["Spectateurs en moyenne", "Spectateurs max.", "Nouveaux followers",
                    "Abonnements payés au total", "Abonnements Prime",
                    "Revenu issu des abonnements", "Revenu Prime", "Minutes passées en stream"]

missing_columns = [col for col in required_columns if col not in data.columns]
if missing_columns:
    print(f"Colonnes manquantes : {missing_columns}")
    exit()

# Liste dynamique des variables disponibles (sauf Date)
available_columns = [col for col in data.columns]
```

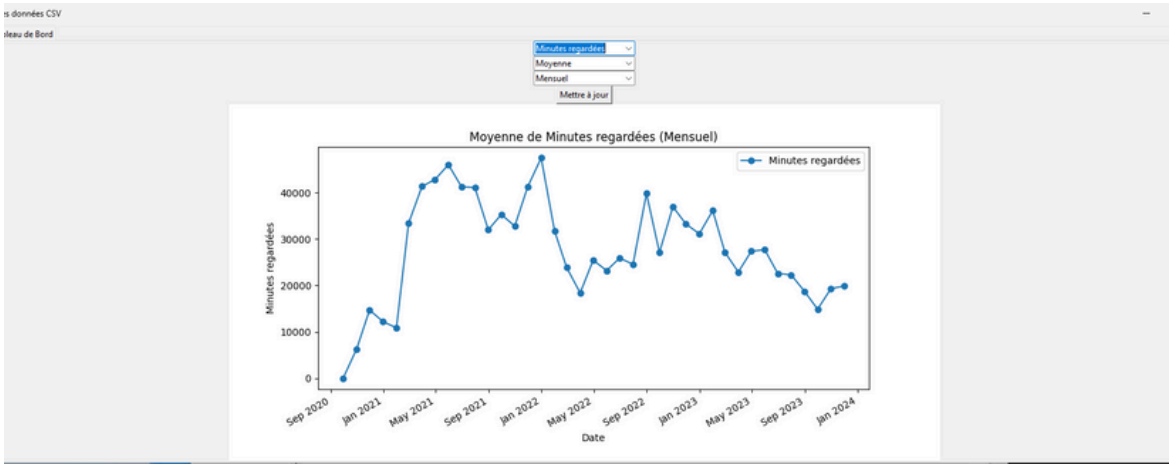
Pour la deuxième partie de notre code python on retrouve la partie du code qui nous permet rafraichir l'interface à chaque click de changement de variables par exemple.

```python
# Fonction pour mettre à jour le graphique
def update_graph():
    ax.clear()
    variable = selected_variable.get()
    stat_tool = selected_stat_tool.get()
    period = selected_period.get()

    freq = {'Hebdomadaire': 'W', 'Mensuel': 'ME', 'Trimestriel': 'QE'}.get(period, 'ME')

    if stat_tool == "Moyenne":
        aggregated_data = data[variable].resample(freq).mean()
    elif stat_tool == "Variance":
        aggregated_data = data[variable].resample(freq).var()
    else:
        aggregated_data = data[variable]

    ax.plot(aggregated_data.index, aggregated_data, marker='o', linestyle='-', label=variable)
    ax.set_title(f"{stat_tool} de {variable} ({period})")
    ax.set_xlabel("Date")
    ax.set_ylabel(variable)
    ax.legend()
    ax.xaxis.set_major_locator(mdates.AutoDateLocator())
    ax.xaxis.set_major_formatter(mdates.DateFormatter('%b %Y'))
    fig.autofmt_xdate()
    canvas.draw()

# Fonction pour mettre à jour le tableau de bord
def update_dashboard():
    for row in treeview.get_children():
        treeview.delete(row)

    num_months = int(selected_dashboard_period.get())

    max_spectateurs = data["Spectateurs max."].resample('ME').max()
    mean_spectateurs = data["Spectateurs en moyenne"].resample('ME').mean().round(2)
    total_followers = data["Nouveaux followers"].resample('ME').sum()
    total_abonnements = data["Abonnements payés au total"].resample('ME').sum()
    total_prime = data["Abonnements Prime"].resample('ME').sum()
    total_revenu_abonnements = data["Revenu issu des abonnements"].resample('ME').sum()
    total_revenu_prime = data["Revenu Prime"].resample('ME').sum()
    total_minutes_stream = data["Minutes passées en stream"].resample('ME').sum()

    max_dates = data.groupby(data.index.to_period('M'))["Spectateurs max."].idxmax()

    last_months = max_spectateurs.index[-num_months:]

    for date in last_months:
        treeview.insert("", "end", values=(
            date.strftime("%Y-%m"),
            mean_spectateurs.loc[date],
            f"{max_spectateurs.loc[date]} ({max_dates.loc[date].strftime('%d-%m-%Y')})",
            total_followers.loc[date],
            total_abonnements.loc[date],
            total_prime.loc[date],
            total_revenu_abonnements.loc[date],
            total_revenu_prime.loc[date],
            total_minutes_stream.loc[date]
        ))
```

Pour la dernière partie de notre code python on retrouve la partie du code la plus importante celle qui nous permet de créer l'interface avec les listes déroulantes les graphiques et notre interface.

```python
# Interface Tkinter
root = tk.Tk()
root.title("Analyse des données CSV")
notebook = ttk.Notebook(root)
notebook.pack(expand=True, fill="both")

# Onglet Graphique
frame_graph = ttk.Frame(notebook)
notebook.add(frame_graph, text="Graphique")

selected_variable = tk.StringVar()
dropdown_variable = ttk.Combobox(frame_graph, textvariable=selected_variable, values=available_columns)
dropdown_variable.pack()

selected_stat_tool = tk.StringVar()
dropdown_stat_tool = ttk.Combobox(frame_graph, textvariable=selected_stat_tool, values=["Moyenne", "Variance"])
dropdown_stat_tool.pack()

selected_period = tk.StringVar()
dropdown_period = ttk.Combobox(frame_graph, textvariable=selected_period, values=["Hebdomadaire", "Mensuel", "Trimestriel"])
dropdown_period.pack()

btn_update_graph = tk.Button(frame_graph, text="Mettre à jour", command=update_graph)
btn_update_graph.pack()

fig, ax = plt.subplots(figsize=(10, 5))
canvas = FigureCanvasTkAgg(fig, master=frame_graph)
canvas.get_tk_widget().pack()

 # Onglet Tableau de Bord
 frame_dashboard = ttk.Frame(notebook)
 notebook.add(frame_dashboard, text="Tableau de Bord")

 selected_dashboard_period = tk.StringVar()
 dropdown_dashboard_period = ttk.Combobox(frame_dashboard, textvariable=selected_dashboard_period, values=["3", "6", "12", "24"])
 dropdown_dashboard_period.pack()

 btn_update_dashboard = tk.Button(frame_dashboard, text="Mettre à jour", command=update_dashboard)
 btn_update_dashboard.pack()

 treeview = ttk.Treeview(frame_dashboard, columns=("Mois", "Moy. Spectateurs", "Max Spectateurs", "Nouv. Followers", "Total Abonnements", "Total Prime",
 "Revenu Abonnements", "Revenu Prime","Minutes Stream"), show="headings")
for col in treeview['columns']:
    treeview.heading(col, text=col)
 treeview.pack(expand=True, fill="both")

 scrollbar = ttk.Scrollbar(frame_dashboard, orient="vertical", command=treeview.yview)
 treeview.configure(yscroll=scrollbar.set)
 scrollbar.pack(side="right", fill="y")

 root.mainloop()
```

Ce qui nous donne ce résultat



CONCLUSION -

Notre projet a permis de développer un outil d'analyse de données Twitch innovant et accessible, répondant aux besoins spécifiques de visualisation et de compréhension des performances de streaming de Sephijin.

Réalisations Principales:

-Création d'une interface graphique intuitive permettant l'analyse de données Twitch sans compétences techniques préalables
-Développement d'un système d'analyse flexible avec des calculs de moyennes et variances sur différentes périodes (hebdomadaire, mensuel, trimestriel)
-Sélection et mise en valeur des indicateurs clés de performance les plus pertinents pour un streamer

Perspectives d'Amélioration:

Bien que notre solution offre déjà une vue d'ensemble précieuse, nous identifions plusieurs axes d'évolution potentiels :

-Intégration de l'analyse des jeux streamés et leur impact sur l'audience
-Ajout de fonctionnalités prédictives basées sur les tendances des jeux
-Développement de mécanismes de comparaison avec d'autres streamers
-Enrichissement de l'interface avec des visualisations de données plus avancées